IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application

Applicant(s):	Hull et al
Case:	5-4-1-4
Serial No.:	09/251,998
Filing Date:	February 19, 1999
Group:	2143
Examiner:	David E England
Title:	Eager Evaluation of Tasks in a Workflow System

REPLY BRIEF

Mail Stop Appeal Brief – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Appellants hereby reply to the Office Action dated October 10, 2006, and to the Examiner's Answer, mailed April 21, 2006 (referred to hereinafter as "the Examiner's Answer"), in an Appeal of the final rejection of claims 1-21 in the above-identified patent application. This response supercedes the response mailed on June 20, 2006

REAL PARTY IN INTEREST

A statement identifying the real party in interest is contained in Appellants' Appeal Brief

RELATED APPEALS AND INTERFERENCES

A statement identifying related appeals is contained in Appellants' Appeal Brief.

STATUS OF CLAIMS

A statement identifying the status of the claims is contained in Appellants' Appeal Brief.

STATUS OF AMENDMENTS

A statement identifying the status of the amendments is contained in Appellants' Appeal Brief.

SUMMARY OF CLAIMED SUBJECT MATTER

A Summary of the Invention is contained in Appellants' Appeal Brief.

STATEMENT OF GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A statement identifying the issues presented for review is contained in Appellants' Appeal Brief.

CLAIMS APPEALED

A copy of the appealed claims is contained in an Appendix of Appellants' Appeal Brief.

ARGUMENT

Rejection Under 35 U.S.C. §112, Second Paragraph

As to Issue (1) presented above, the Examiner rejected claims 1-21 under 35 U.S.C. §112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. In particular, the Examiner asserts that it is unclear as to "what/where specifically the execution of the task resulting in an initiation of a side-effect action is included in the eligibility for eager execution." Appellants respectfully traverse this rejection.

Appellants note that the cited claims clearly indicate that whether a task is eligible for eager execution is determined by considering whether *execution of the task results in the initiation of a side-effect action*. The specification clearly teaches that "the execution of at least some of the tasks (so-called side-effect tasks) results in the initiation

of a side-effect action *performed by a component external to the workflow system*" (Page 2, lines 26-29, of the originally filed specification; emphasis added.) Thus, the definition of a side-effect action and the utilization of such an event to determine whether a task is eligible for eager execution are clearly defined in the disclosure

Rejection under 35 U.S.C. §102(e) Over U.S. Patent No. 6,697,935

The Examiner rejected claims 1-3, 5, 9, 12-14 and 16 under 35 U.S.C. §102(e) as being anticipated by Borkenhagen.

Claims 1 and 12

Regarding independent claims 1 and 12, the Examiner asserts that Borkenhagen teaches determining whether a task is eligible for eager execution by considering . . .whether execution of the task results in the initiation of a side-effect action (e.g., col. 18, lines 37-51).

Appellants note that Borkenhagen is directed to "a multithreaded processor capable of switching execution between two threads of instructions, and thread switch logic embodied in hardware registers with optional software override of thread switch conditions." (Col. 5, lines 7-11.) In the text cited by the Examiner, Borkenhagen teaches "the priorities of the threads can be adjusted by the thread switch manager software through the use of one or more instructions, or *by hardware in response to an* ***event***." (Col. 18, lines 46-48; emphasis added.) Independent claims 1 and 12 require determining whether a task is eligible for eager execution by considering at least (1) a state of the task and (2) whether *execution of the task results in the* ***initiation of a side-effect action***. The present invention considers whether the execution of a task **initiates** a side-effect action (defined in the specification as being performed by a component ***external*** to the workflow system), whereas Borkenhagen adjusts the priorities of threads in **response to** instructions or events. Borkenhagen does not disclose or suggest considering whether the execution of a task results in a side-action being performed by a component ***external*** to the system.

Thus, Borkenhagen does not disclose or suggest determining whether a task is eligible for eager execution by considering at least (1) a state of the task and (2)

whether execution of the task results in the initiation of a side-effect action, as required by independent claims 1 and 12.

Claims 2 and 13

With regard to claims 2 and 13, which stand or fall together, these claims contain the limitations of "determining that a particular task whose execution results in the initiation of a side-effect action is eligible for eager execution only if it is determined that the one or more enabling conditions associated with the particular task will evaluate to true as determined by the state of the particular task."

Appellants respectfully submit that, while Borkenhagen does describe a number of states for a task, there is no teaching in Borkenhagen of determining that one or more enabling conditions associated with the particular task will evaluate to true as determined by the state of the particular task. Appellants define enabling conditions in independent claims 1 and 12 (from which dependent claims 2 and 13, respectively, depend) as "one or more of said tasks each having one or more associated enabling conditions indicating whether the task is to be executed for said object." See also page 2, lines 24-26 of the present specification. There is no determination that an enabling condition for a task in Borkenhagen will evaluate to true as determined by the state of the task. Even if Borkenhagen does determine that an enabling condition for a task will evaluate to true, this determination is not made by using state of the task in Borkenhagen.

There is no disclosure in Borkenhagen of tasks that have enabling conditions as defined by and used in independent claims 1 and 12 and dependent claims 2 and 13. Consequently, Appellants respectfully submit that dependent claims 2 and 13 are patentable over Borkenhagen, alone or in combination.

Claims 3 and 14

With regard to claims 3 and 14, which stand or fall together, each of these claims has the limitations of "determining that a particular task whose execution does not result in the initiation of a side-effect action is eligible for eager execution prior to determining that the one or more enabling conditions associated with the particular task will evaluate to true as determined by the state of the particular task."

Appellants can find no disclosure in Borkenhagen that Borkenhagen determines that a task is eligible for eager execution prior to determining that an

associated enabling condition will evaluate to true, as claimed in dependent claims 3 and 14. Consequently, Appellants respectfully submit that dependent claims 3 and 14 are patentable over Borkenhagen.

Claims 5 and 16

With regard to claims 5 and 16, which stand or fall together, these claims have the additional limitations of "wherein said step of determining whether a task is eligible for eager execution is performed by also considering (3) whether the task contributes to the production of a target value." The Examiner asserts that Borkenhagen discloses this limitation (col. 3, line 55, to col 4, line 6).

Appellants respectfully disagree. Appellants could find no disclosure or suggestion in Borkenhagen to determine whether a task contributes to the production of a target value. In fact, Appellants could find no disclosure or suggestion by Borkenhagen of a target value.

Consequently, Appellants respectfully submit that dependent claims 5 and 16 are patentable over Borkenhagen, alone or in combination.

Rejection under 35 U.S.C. §103(a) Over U.S. Patent No. 6,697,935 in View of U.S. Patent No. 6,253,307

With regard to Issue (3), the Examiner rejected claims 4, 6, 7, 8, 15 and 17-19 under 35 U.S.C. §103(a) as being unpatentable over Borkenhagen in view of Boutaud.

Claims 4 and 15

With regard to claims 4 and 15, which stand or fall together, these claims add the limitation of "partially evaluating one or more enabling conditions associated with said task." The Examiner points to col. 45, line 58, to col. 46, line 51, of Boutaud as teaching this limitation. Appellants read the cited sections of Boutaud as describing "conditional instructions" that can be or not be executed based on a condition. See, for instance, col. 46, lines 13-25, of Boutaud. Appellants respectfully submit that the cited text of Boutaud does not disclose any item that is partially evaluated. For example, if a condition in Boutaud is true, then certain conditional instructions are executed; if a

condition in Boutaud is false, then those certain conditional instructions are not executed See col. 46, lines 20-25, of Boutaud.

By contrast, an enabling condition of the present invention can be partially evaluated For example, FIG. 20 shows the enabling condition "cust_value < 7 and DNIS not = 'Australia_promotion '" The "cust_value < 7" part of the enabling condition could be evaluated, which means that only part of the enabling condition "cust_value < 7 and DNIS not = 'Australia_promotion'" would be evaluated. In Boutaud, the conditional instructions are either executed or not executed, and there is no partial evaluation of the conditional instructions

Thus, Appellants respectfully submit that independent claims 4 and 15 are patentable over Borkenhagen and Boutaud, alone or in combination

Claims 6 and 17

With regard to claims 6 and 17, which stand or fall together, these claims have the limitation of "determining that a particular task is unneeded for processing of the object based at least in part on partial evaluation of an enabling condition of a second task, wherein said second task's enabling condition depends on one or more outputs of said particular task" As described above, Appellants submit that Boutaud does not disclose a partial evaluation of an enabling condition Moreover, in Boutaud if a "conditional instruction" is considered to be a "task," then there is no determination that a particular conditional instruction (i.e., "task") is necessary based on evaluation of an enabling condition of a second conditional instruction (i.e., "task") Instead, in Boutaud, if a condition is or is not true, the "conditional instructions" are or are not executed, respectively, and conditional instructions do not depend on enabling conditions of other conditional instructions Consequently, Appellants respectfully submit that dependent claims 6 and 17 are patentable over Borkenhagen and Boutaud

Claims 7 and 18

Regarding claims 7 and 18, which stand or fall together, the same reasoning applies as in regards to dependent claims 6 and 17. Furthermore, if a "conditional instruction" in Boutaud is considered to be a "task," Boutaud does not disclose that a particular conditional instruction (e.g., "task") is based on evaluation of enabling conditions for a number of tasks, as conditional instructions in Boutaud either

are or are not executed depending on a single condition Consequently, Appellants respectfully submit that dependent claims 7 and 18 are patentable over Borkenhagen and Boutaud.

Claims 8 and 19

Regarding claims 8 and 19, which stand or fall together, these claims add the limitations of "determining that a particular task is necessary for processing of the object based at least in part on evaluation of enabling conditions for a number of tasks, wherein said tasks' enabling conditions depend on one or more outputs of said particular task." Appellants submit that these claims are patentable for at least the reasons stated above with regard to claims 5, 6, 16 and 17. Furthermore, if a "conditional instruction" in Boutaud is considered to be a "task," then Boutaud does not disclose that a number of conditional instructions' (i.e., "tasks"") enabling conditions depend on outputs of a particular conditional instruction (i e , "task"), as conditional instructions in Boutaud either are or are not executed depending on a single condition Consequently, Appellants respectfully submit that dependent claims 8 and 19 are patentable over Borkenhagen and Boutaud.

Rejection under 35 U.S.C. §103(a) Over U.S. Patent No. 6,697,935 in View of U.S. Patent No. 5,854,929 and U.S. Patent No. 5,561,762

With regard to Issue (4), the Examiner rejected claims 10, 11, 20, and 21 under 35 U.S.C. §103(a) as being unpatentable over Borkenhagen in view of Van Praet and Smith.

With regard to claims 10, 11, 20, and 21, which stand or fall together, each of claims 10 and 20 has the limitations of "a graph representing data flow dependencies and enabling flow dependencies between tasks and enabling conditions" and "propagating changes through said graph based on new outputs of completed tasks." Claim 11 depends from claim 10 and claim 21 depends from claim 20. Van Praet discloses a "bipartite" graph where vertices represent storage elements in a processor or operations of a processor, and where edges represent connectivity of a processor and data flow from storage. See, col. 8, lines 51-57 of Van Praet Smith discloses a graph where

each node represents a logic gate and the branches represent input or output lines. See, col. 5, lines 51-58, of Smith.

In the present invention, as described above, a task has one or more associated enabling conditions indicating whether the task is to be executed for an object (see, e.g., independent claims 1 and 12). Furthermore, a task can produce an output that is used in an enabling condition for another task. See, for instance, FIG. 26 and associated text on pages 36 and 37 of the present specification, where it states the following:

> this diagram illustrates the data flow dependencies and the enabling flow dependencies of the workflow described above. Each of the modules (ovals) and enabling conditions (hexagons) are represented as nodes with solid line data flow edges representing data flow dependencies and broken line enabling flow edges representing enabling flow dependencies.

If a "task" of the present invention is a store element or processor of Van Praet, while Van Praet might, for sake of argument, show a data flow dependency in a graph, there is no disclosure of an enabling flow dependency in the graph. Similarly, if a "task" of the present invention is a logic gate, while Smith might, for sake of argument, show a data flow dependency in a graph, there is no disclosure of an enabling flow dependency in the graph. In other words, in both Van Praet and Smith, only one data dependency (e.g., "edge" or "connection") is shown between nodes, while claims 10 and 20 require two types of data dependencies. Consequently, Appellants respectfully submit that dependent claims 10 and 20 are patentable over Borkenhagen, Van Praet, and Smith, alone or in combination. Because claims 10 and 20 are patentable, their respective dependent claims 11 and 21 are patentable.

Rejection under 37 CFR 1.83(a)

The drawings were objected to under 37 CFR 1.83(a) for not showing every feature of the invention specified in the claims. In particular, the Examiner asserts that the "determining whether a task is eligible for eager execution by considering at least (1) a state of the task and (2) whether execution of the task results in the initiation of a side-effect action" must be shown in the drawings.

Appellants note that the limitation cited by the Examiner is shown in a multitude of drawings, including FIGS. 4, 6-8, 10-25 (see, enabling condition, side effect, and side effect function), 29, 30 (blocks 3022 and 3026), 36, and 37. Appellants believe that every feature of the invention specified in the claims is properly shown in the drawings.

Response to Examiner's Answer of November 3, 2005 & April 21, 2006

First Argument

In response to Applicant's first argument, the Examiner asserts that, in the cited area of the specification given by the Applicant, there is no disclosure of where the task is being executed in the workflow system or what part of the workflow system is performing the processing of the task, and that whether or not it has a side-effect action is not the question nor where the side-effect action occurs.

Appellants note that where the task is being executed in the workflow system or what part of the workflow system performs the processing of the task is a design choice, as would be apparent to a person of ordinary skill in the art.

Second Argument

In response to Applicant's second argument, the Examiner asserts that, since it is not explicitly stated in the independent claims what would specifically constitute a "side-effect action," one would interpret as broadly as possible. Appellants maintain, however, that, by definition, an action initiated by the execution of a task could only be considered a ***side-effect*** (of the task execution) if it were a result of the task execution; an action initiated by an ***external*** *component or event* could not be considered a side-effect of the task execution.

Third Argument

In response to Applicant's third argument, the Examiner asserts that the switch control register 410, which can be interpreted as "one or more enabling conditions" has a value of one. Appellants note that, regarding switch control register 410, Borkenhagen teaches that

> thread switch logic 400 comprises a thread switch control register 410 which *controls what events will result in a thread switch*. For instance, the thread switch control register 410 can *block events that cause*

> *state changes from being seen by the thread switch controller 450 so that a thread may not be switched as a result of a blocked event. The thread state registers and the logic of changing threads are the subject of a U.S. patent application Ser. No. 08/957,002*, filed concurrently and herein incorporated by reference. The forward progress count register 420 is used to prevent thrashing and may be included in the thread switch control register 410.
> (Col. 10, lines 39-50; emphasis added)

Borkenhagen also teaches that

> the thread switch control register 410 is a software programmable register which *selects the events to generate thread switching and has a separate enable bit for each defined thread switch control event*. Although the embodiment described herein does not implement a separate thread switch control register 410 for each thread, separate thread switch control registers 410 for each thread could be implemented to provide more flexibility and performance at the cost of more hardware and complexity. Moreover, the thread switch control events in one thread switch control register need not be identical to the thread switch control events in any other thread switch control register.
>
> The thread switch control register 410 can be written by a service processor with software such as a dynamic scan communications interface disclosed in U.S. Pat. No. 5,079,725 entitled Chip Identification Method for Use with Scan Design Systems and Scan Testing Techniques or by the processor itself with software system code. The contents of the thread switch control register 410 is used by the thread switch controller 450 *to enable or disable the generation of a thread switch*. A value of one in the register 410 enables the thread switch control event associated with that bit to generate a thread switch. A value of zero in the thread switch control register 410 disables the thread switch control event associated with that bit from generating a thread switch. Of course, an instruction in the executing thread could disable any or all of the thread switch conditions for that particular or for other threads. The following table shows the association between thread switch events and their enable bits in the register 410.
> (Col. 14, lines 3-30; emphasis added)

Appellants maintain that the value stored in a switch control register *is **not** a state of a task* and is ***not** an enabling condition,* as defined in claims 1 and 12 and page 2 of the specification (see, Appellants argument regarding claims 2 and 13 in the Appeal Brief).

Fourth Argument

In response to Applicant's fourth argument, the Examiner asserts that the Applicant's argument fails to comply with 37 CFR 1.111(b) because they amount to a

general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references. Appellants maintain, however, that the Examiner's citation in the Examiner's Answer (regarding Appellant's argument for claims 3 and 14) specifically points out how the language of the claims patentably distinguishes them from the references since claims 3 and 14 recite this language. More specifically, in the Appeal Brief, Appellants argued that no disclosure or suggestion could be found "in Borkenhagen that Borkenhagen determines that a task is eligible for eager execution prior to determining that an associated enabling condition will evaluate to true, as claimed in dependent claims 3 and 14." This argument specifically points out how the language of the claims patentably distinguishes the cited claims from the references.

Fifth Argument

In response to Applicant's fifth argument, the Examiner asserts that the Applicant's argument fails to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references. Appellants maintain, however, that the Examiner's citation in the Examiner's Answer specifically points out how the language of the claims patentably distinguishes them from the references since claims 5 and 16 recite this language. More specifically, in the Appeal Brief, Appellants argued that no disclosure or suggestion could be found "in Borkenhagen to determine whether a task contributes to the production of a target value. In fact, Appellants could find no disclosure or suggestion by Borkenhagen of a target value." This argument specifically points out how the language of the claims patentably distinguishes the cited claims from the references.

Sixth and Seventh Arguments

In response to Applicant's sixth and seventh arguments, the Examiner asserts that the teaching of Boutaud in regard to an "IF THEN ELSE" condition code reads on what could be interpreted as "partially evaluating." Appellants note that, contrary to the Examiner's assertion, an "IF THEN ELSE" condition implies that the entire "IF" condition is evaluated to determine whether the "THEN" or "ELSE" actions are executed. Boutaud does not disclose or suggest evaluating only a part of the "IF"

condition As the Examiner notes, the present disclosure teaches that only a portion of a condition (e g , "cust value < 7") may be evaluated.

Eighth Argument

In response to Applicant's eighth argument (labeled by the Examiner as the (second) seventh argument), the Examiner asserts that, "in Van Praet, it is taught in figure 5 and column 5, lines 53 et seq , graphing data flow dependencies and enabling flow dependencies as data flow graph (DFG) and instruction set graph (ISG), ('In this way the same relations are obtained as depicted in Fig B, but with DFG and an ISG of much lower complexity') As to further support the teachings of Van Praet, the ISG it taught to have a set of instructions that enables an operation I in the ISG is called its enabling condition and denoted by enabling(i), (e.g., col 8, lines 64 - col 9, line 15) "

As noted in the Appeal Brief, Van Praet discloses a "bipartite" graph where vertices represent storage elements in a processor or operations of a processor, and where edges represent connectivity of a processor and data flow from storage. See col. 8, lines 51-57 of Van Praet Smith discloses a graph where each node represents a logic gate and the branches represent input or output lines. See col. 5, lines 51-58 of Smith.

In the present invention, as described above, a task has one or more associated enabling conditions indicating whether the task is to be executed for an object (see, e.g., independent claims 1 and 12). Furthermore, a task can produce an output that is used in an enabling condition for another task. See, for instance, FIG. 26 and associated text on pages 36 and 37 of the present specification, where it states the following:

> this diagram illustrates the data flow dependencies and the enabling flow dependencies of the workflow described above. Each of the modules (ovals) and enabling conditions (hexagons) are represented as nodes with solid line data flow edges representing data flow dependencies and broken line enabling flow edges representing enabling flow dependencies.

If a "task" of the present invention is a store element or processor of Van Praet, while Van Praet might, for sake of argument, show a data flow dependency in a graph, there is no disclosure of an enabling flow dependency in the graph. Similarly, if a "task" of the present invention is a logic gate, while Smith might, for sake of argument,

show a data flow dependency in a graph, there is no disclosure of an enabling flow dependency in the graph. In other words, in both Van Praet and Smith, only one data dependency (e.g., "edge" or "connection") is shown between nodes, while claims 10 and 20 require two types of data dependencies.

Conclusion

The remaining rejected dependent claims are believed allowable for at least the reasons identified above with respect to the independent claims. Appellants respectfully submit that claims 1-21 of the present invention are patentable.

The attention of the Examiner and the Appeal Board to this matter is appreciated

Respectfully,

Date: November 1, 2006

Kevin M. Mason
Attorney for Applicant(s)
Reg. No. 36,597
Ryan, Mason & Lewis, LLP
1300 Post Road, Suite 205
Fairfield, CT 06430
(203) 255-6560

APPENDIX

1. A method for operation of a workflow system for processing an object by executing a plurality of tasks, one or more of said tasks each having one or more associated enabling conditions indicating whether the task is to be executed for said object, and wherein execution of at least one of said tasks results in initiation of a side-effect action performed by a component external to said workflow system, said method comprising the steps of:

determining whether a task is eligible for eager execution by considering at least (1) a state of the task and (2) whether execution of the task results in the initiation of a side-effect action; and

executing the task using eager execution if the task is determined to be eligible for eager execution.

2. The method of claim 1 wherein the step of determining whether a task is eligible for eager execution further comprises the step of:

determining that a particular task whose execution results in the initiation of a side-effect action is eligible for eager execution only if it is determined that the one or more enabling conditions associated with the particular task will evaluate to true as determined by the state of the particular task.

3 The method of claim 1 wherein the step of determining whether a task is eligible for eager execution further comprises the step of:

determining that a particular task whose execution does not result in the initiation of a side-effect action is eligible for eager execution prior to determining that the one or more enabling conditions associated with the particular task will evaluate to true as determined by the state of the particular task

4 The method of claim 1 wherein said step of determining whether a task is eligible for eager execution further comprises the step of:

partially evaluating one or more enabling conditions associated with said task.

5. The method of claim 1 wherein said step of determining whether a task is eligible for eager execution is performed by also considering (3) whether the task contributes to the production of a target value.

6. The method of claim 1 further comprising the step of:

determining that a particular task is unneeded for processing of the object based at least in part on partial evaluation of an enabling condition of a second task, wherein said second task's enabling condition depends on one or more outputs of said particular task.

7 The method of claim 1 further comprising the step of:

determining that a particular task is necessary for processing of the object based at least in part on evaluation of enabling conditions for a number of tasks, wherein said tasks' enabling conditions depend on said particular task

8. The method of claim 1 further comprising the step of:

determining that a particular task is necessary for processing of the object based at least in part on evaluation of enabling conditions for a number of tasks, wherein said tasks' enabling conditions depend on one or more outputs of said particular task.

9 The method of claim 1 wherein said step of determining is performed repeatedly during the processing of the object.

10. The method of claim 1 wherein a memory of said workflow system stores a graph representing data flow dependencies and enabling flow dependencies between tasks and enabling conditions, said method further comprising the step of:

propagating changes through said graph based on new outputs of completed tasks.

11. The method of claim 10 wherein said step of propagating changes is based on predefined propagation rules

12. A workflow system for processing an object by executing a plurality of tasks, one or more of said tasks each having one or more associated enabling conditions indicating whether the task is to be executed for said the object, and wherein execution of at least one of said tasks results in initiation of a side-effect action performed by a component external to said workflow system, said system comprising:

means for determining whether a task is eligible for eager execution by considering at least (1) a state of the task and (2) whether execution of the task results in the initiation of a side-effect action; and

means for executing the task using eager execution if the task is determined to be eligible for eager execution.

13. The workflow system of claim 12 wherein the means for determining whether a task is eligible for eager execution further comprises:

means for determining that a particular task whose execution results in the initiation of a side-effect action is eligible for eager execution only if it is determined that the one or more enabling conditions associated with the particular task will evaluate to true as determined by the state of the particular task.

14. The workflow system of claim 12 wherein the means for determining whether a task is eligible for eager execution further comprises:

means for determining that a particular task whose execution does not result in the initiation of a side-effect action is eligible for eager execution prior to determining that one or more enabling conditions associated with the particular task will evaluate to true as determined by the state of the particular task.

15. The workflow system of claim 12 wherein said means for determining whether a task is eligible for eager execution further comprises:

means for partially evaluating one or more enabling conditions associated with said task.

16. The workflow system of claim 12 wherein said means for determining whether a task is eligible for eager execution further comprises:

means for determining whether the task contributes to the production of a target value.

17. The workflow system of claim 12 further comprising:

means for determining that a particular task is unneeded for processing of the object based at least in part on partial evaluation of an enabling condition of a second task, wherein said second task's enabling condition depends on one or more outputs of said particular task.

18. The workflow system of claim 12 further comprising:

means for determining that a particular task is necessary for processing of the object based at least in part on evaluation of enabling conditions for a number of tasks, wherein said tasks' enabling conditions depend on said particular task.

19. The workflow system of claim 12 further comprising:

means for determining that a particular task is necessary for processing of the object based at least in part on evaluation of enabling conditions for a number of tasks, wherein said tasks' enabling conditions depend on one or more outputs of said particular task.

20. The workflow system of claim 12 further comprising:

a memory for storing a graph representing data flow dependencies and enabling flow dependencies between tasks and enabling conditions; and

means for propagating changes through said graph based on new outputs of completed tasks.

21. The workflow system of claim 20 wherein said memory stores predefined propagation rules and wherein said means for propagating changes further comprises means for propagating changes based on said predefined propagation rules.

EVIDENCE APPENDIX

There is no evidence submitted pursuant to § 1.130, 1.131, or 1.132 or entered by the Examiner and relied upon by appellant.

RELATED PROCEEDINGS APPENDIX

There are no known decisions rendered by a court or the Board in any proceeding identified pursuant to paragraph (c)(1)(ii) of 37 CFR 41.37.

5